(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 690 370 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.01.1996 Bulletin 1996/01

(51) Int Cl.[6]: **G06F 9/38**, G06F 13/16

(21) Application number: **95304221.5**

(22) Date of filing: **19.06.1995**

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **30.06.1994 IL 11018194**

(71) Applicant: **SOFTCHIP ISRAEL LTD.**
Jerusalem 97272 (IL)

(72) Inventor: **Cohen, Michael**
Jerusalem 97225 (IL)

(74) Representative: **Hillier, Peter et al**
London, WC2A 1QU (GB)

(54) Microprocessor device and peripherals

(57) This invention discloses a microprocessor device comprising an execution unit which is governed by a master clock signal and a repeatedly programmable master clock divider which is operative to divide the master clock signal. The microprocessor device is operative to execute instructions executed by an existing microprocessor, wherein said instructions being executed in a given number of machine cycles. Furthermore, the microprocessor includes a programmable instruction mapper containing a multiplicity of mapping schemes for controlling the microprocessor apparatus. A tamper-resistant execution device is included operative to execute instructions arriving from an instruction register. The microprocessor device also includes a pseudo-random access delay signal generator which generates a signal for determining the number of clock cycles elapsing from a memory access instruction cycle to actual memory access. Additionally, a method is also included for testing the contents and integrity of a condition signal within a single machine cycle.

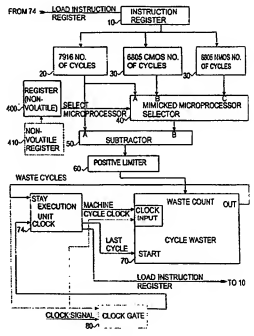


Fig 1

EP 0 690 370 A2

Description

FIELD OF THE INVENTION

The present invention relates to microprocessors.

BACKGROUND OF THE INVENTION

The datasheet of the DS80C320 High Speed Micro, commercially available from Dallas Semiconductor, states that the DS80C320 has a stretch memory cycle feature which (p. 9) "allows the application software to adjust the speed of data memory access. The micro is capable of performing the MOVX in as little as two instruction cycles. However, this value can be stretched as needed so that both fast memory and slow memory or peripherals can be accessed with no glue logic."

However, the stretch memory cycle feature still does not provide timing compatibility of the DS80C320 to existing devices such as the Intel 80C32 device, as specifically stated on p. 4 of the data sheet: "The exception is critical timing since the High Speed Micro performs its instructions much faster than the original."

Published French Patent Application No. 2667715 (90 12440) describes a card which receives data or commands and transmits an "end of command" or "reception acknowledgement" signal after a time T of random duration which does not indicate the type of command performed by the card.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a microprocessor which is an optimized and faster version of an existing microprocessor and which is selectably timing compatible therewith.

It is also an object of the present invention to provide a method for causing a first microprocessor which is optimized and faster than a second microprocessor to become timing compatible with the second microprocessor.

There is thus provided, in accordance with a preferred embodiment of the present invention, a microprocessor device including an execution unit which is governed by a master clock signal, and a repeatedly programmable master clock divider which is operative to divide the master clock signal.

Also provided, in accordance with another preferred embodiment of the present invention, is a microprocessor device operative to execute the instructions $I_1, ..., I_k$ executed by an existing microprocessor, the instructions being executed by the existing microprocessor in $n_1, ..., n_k$ machine cycles respectively, the microprocessor device including an execution unit operative to execute the plurality of instructions in $m_1, ..., m_k$ machine cycles respectively, where $m_i <= n_i$ for $i = 1, ..., k$, and a machine cycle controller operative in response to a "select microprocessor" signal, to waste machine cycles such that, for each instruction, the same number of machine cycles are expended by the execution unit as by the existing microprocessor.

Further in accordance with a preferred embodiment of the present invention, the programmable master clock divider is one-time programmable.

Still further in accordance with a preferred embodiment of the present invention, the programmable master clock divider is execution-time programmable.

Additionally in accordance with a preferred embodiment of the present invention, the machine cycle controller is one-time programmable.

Further in accordance with a preferred embodiment of the present invention, the machine cycle controller is execution-time programmable.

Still further in accordance with a preferred embodiment of the present invention, the execution unit includes a state machine.

Additionally in accordance with a preferred embodiment of the present invention, the execution unit includes a microcode controller and associated microcode.

Further in accordance with a preferred embodiment of the present invention, the machine cycle controller is operative to instruct the execution unit to remain in its current state until the same number of machine cycles have been expended as by the existing microprocessor.

Still further in accordance with a preferred embodiment of the present invention, the device also includes a clock gate and the machine cycle controller is operative to instruct the clock gate to block the clock to the execution unit.

There is also provided, in accordance with another preferred embodiment of the present invention, microprocessor control apparatus including an instruction register receiving an individual one of a plurality of instruction codes, an intrinsic instruction execution unit containing a plurality of intrinsic instruction functions and an input, the unit being operative to receive a control signal through the input which designates an individual one of the intrinsic instruction functions and to execute the designated intrinsic instruction function, and a programmable instruction mapper containing a multiplicity

of mapping schemes and being operative to employ a selected scheme to convert the instruction code in the register into an intrinsic instruction designation which is provided as a control signal to the input of the execution unit.

Further in accordance with a preferred embodiment of the present invention, the control signal is stored in a register which is loaded by a specific instruction performed by the execution unit.

Still further in accordance with a preferred embodiment of the present invention, the control signal is stored in a register which is mapped as a memory location such that the control signal is introduced therein by writing.

Further in accordance with a preferred embodiment of the present invention, the control signal is stored in a non-volatile register which is loaded by programming.

Still further in accordance with a preferred embodiment of the present invention, the apparatus includes a volatile register storing the control signal, and a non-volatile register storing a default value which is copied into the volatile register.

Further in accordance with a preferred embodiment of the present invention, the apparatus also includes a register containing a control signal for the clock divider, wherein the register is loaded by a specific instruction performed by the execution unit.

Still further in accordance with a preferred embodiment of the present invention, the apparatus also includes a register containing a control signal for the clock divider, wherein the register is mapped as a memory location such that the control signal is introduced therein by writing.

Further in accordance with a preferred embodiment of the present invention, the apparatus also includes a non-volatile register which contains a control signal for the clock divider and which is loaded by programming.

Still further in accordance with a preferred embodiment of the present invention, the apparatus also includes a volatile register storing a control signal for the clock divider, and a non-volatile register storing a default value which is copied into the volatile register.

Further in accordance with a preferred embodiment of the present invention, the "select microprocessor" signal is stored in a register which is loaded by a specific instruction performed by the execution unit.

Still further in accordance with a preferred embodiment of the present invention, the "select microprocessor" signal is stored in a register which is mapped as a memory location such that the "select microprocessor" signal is introduced therein by writing.

Additionally in accordance with a preferred embodiment of the present invention, the "select microprocessor" signal is stored in a non-volatile register which is loaded by programming.

Further in accordance with a preferred embodiment of the present invention, the apparatus also includes a volatile register storing the "select microprocessor" signal, and a non-volatile register storing a default value which is copied into the volatile register.

There is also provided, in accordance with another preferred embodiment of the present invention, a tamper-resistant execution device operative to execute instructions arriving from an instruction register, the device including an execution unit receiving and carrying out instructions received from the instruction register, a tamper attempt detector operative to detect attempts to tamper with the device, and an instruction replacement module intermediate the instruction register and the execution unit and operative, upon receipt of a "tamper attempt detected" input from the detector, to override instructions arriving from the instruction register with other instructions.

There is further provided, in accordance with another preferred embodiment of the present invention, a method for performing a robust conditional jump including testing the contents and the integrity of a condition signal within a single machine cycle, jumping to a first location if the condition signal indicates that the integrity of the condition is not maintained, jumping to a second location if the condition signal indicates that the integrity of the condition is maintained and the condition is fulfilled, and jumping to a third location if the condition signal indicates that the integrity of the condition is maintained and the condition is not fulfilled.

There is further provided, in accordance with another preferred embodiment of the present invention, memory access apparatus including a pseudo-random access delay signal generator generating an access delay signal which is at least pseudo-random and which determines the number of clock cycles to elapse from a memory access instruction clock cycle to actual memory access, and a memory access unit receiving the access delay signal and operative to access the memory after the number of clock cycles has elapsed.

Further in accordance with a preferred embodiment of the present invention, the pseudo-random signal generator includes a random signal generator.

Still further in accordance with a preferred embodiment of the present invention, the number of clock cycles allowed for memory access, once memory access has initiated, is programmable.

Further in accordance with a preferred embodiment of the present invention, the range of the random/pseudo-random access delay signal is programmable.

Still further in accordance with a preferred embodiment of the present invention, the number of cycles allocated to machine memory access is programmable.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood and appreciated from the following detailed description, taken in conjunction with the drawings in which:

Fig. 1 is a simplified functional block diagram of a peripheral unit for a microprocessor constructed and operative in accordance with a preferred embodiment of the present invention; and

Fig. 2 is a simplified functional block diagram of a more optimized variation of the apparatus of Fig. 1.

Fig. 3 is a simplified block diagram of a peripheral unit which is a modification of the apparatus of Fig. 2, having a repeatedly programmable master clock divider;

Fig. 4 is a simplified block diagram of a microprocessor constructed and operative in accordance with a preferred embodiment of the present invention having the programmable clock division feature of Fig. 3, without the cycle wasting feature of Fig. 2;

Fig. 5 is a simplified block diagram of microprocessor control apparatus employing a selected mapping scheme, which is constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 6 is a simplified block diagram of microprocessor control apparatus which includes a tamper-resistant execution module 360;

Fig. 7 is a simplified block diagram of a jump unit constructed and operative in accordance with a preferred embodiment of the present invention which is operative to perform a robust conditional jump;

Fig. 8A is a prior art timing diagram for access to memory;

Fig. 8B is a preferred timing diagram, provided in accordance with a preferred embodiment of the present invention, for access to memory; and

Fig. 9 is a very generalized prior art block diagram of a microprocessor which is useful in understanding the notations of Figs. 8A and 8B.

Attached herewith is the following appendix which aid in the understanding and appreciation of one preferred embodiment of the invention shown and described herein:

Appendix A is a listing of a suitable program which performs the functions of the tables and selector of Fig. 2.

Alternative or optional implementations of the present invention are illustrated in the drawings by a "dotted line".

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Reference is now made to Fig. 1 which is a simplified functional block diagram of a peripheral unit for a microprocessor device, also termed herein "the 7916 device", which interacts with the execution unit of the 7916 device. The peripheral unit of Fig. 1 enables the 7916 device to mimic an existing microprocessor device, or alternatively a selected one of a plurality of existing microprocessor devices, the peripheral unit being constructed and operative in accordance with a preferred embodiment of the present invention.

The apparatus of Fig. 1 includes a conventional instruction register 10 feeding into a plurality of tables including a first table 20 storing the number of machine cycles required for each instruction by the microprocessor device itself. The plurality of tables also includes one or more second tables 30 which store the number of machine cycles required for each instruction by other existing microprocessor devices or microprocessor device families which perform at least some of the same instructions as does the microprocessor of Fig. 1. In the illustrated embodiment, the existing microprocessors which are supported are the Motorola 6805 CMOS microprocessor family and the Motorola 6805 NMOS microprocessor family.

In each table, the number of machine cycles stored for an instruction which the microprocessor device described by that table does not perform, is zero.

The tables are addressed by the instruction register 10 which indicates the instruction currently being executed.

Selection of the mimicked device is performed by a multiplexer 40 which receives a control signal. The multiplexer 40 selects the output of the table representing the device to be mimicked based on the control signal and feeds this

data into a subtractor 50 which subtracts from this value the number of machine cycles of the 7916 device arriving directly from table 20.

The output of the subtractor is provided to a positive limiter 60 which zeros input values which are negative.

The output of the limiter 60 is provided to a cycle waster 70 which receives three control inputs. The first control input indicates the last cycle of the current instruction. The second control input indicates the machine cycle clock. Cycle wasting is counted by an internal counter, "count". The third control input, waste count, indicates the number of cycles to wait. Upon each wasted cycle, typically during the rising edge of the "machine cycle clock" control signal, "count" is decremented. Cycle wasting is initiated upon receipt of the "last cycle" input and continues until "count" reaches 0. The output of the cycle waster 70 is a "waste cycles" signal.

A preferred mode of operation for the cycle waster 70 is as follows:

```
IF (STARTED AND CLOCK) (
    IF (COUNT <> 0) (
        OUT = WASTE_COUNT
        COUNT = COUNT - 1
    )ELSE (
    STARTED = FALSE
        OUT = CONTINUE
    )
)
```

Preferably, the "waste cycles" signal instructs the execution unit 74 of the 7916 device, typically comprising a state machine or a microcode controller and associated microcode, to remain in its current state. Alternatively, a clock gate 80 may be provided and the "waste cycle" signal may instruct the clock gate 80 to block the clock to the execution unit, in which case the clock signal is fed directly into the cycle waster 70 and the tables 20 and 30 store the number of clock cycles required for each instruction. This alternate implementation is illustrated by the "dotted line" in Figs. 1 and 2.

According to still a further alternative, the "waste cycles" signal may instruct the execution unit to perform one or more operations which have no net effect. For example, the execution unit may be instructed to carry out a NOP operation.

It is appreciated that the apparatus of Fig. 1 may be retrofitted to a conventional microprocessor device such as, for example, the DS80C320 which is an optimized faster version of the Intel 80C32, thereby to render the conventional microprocessor device capable of mimicking other existing devices. For example, the DS80C320 may, by virtue of the present invention, be rendered capable of mimicking the Intel 80C32 whereas, currently, the DS80C320 is incapable of mimicking the Intel 80C32 in timing-critical situations.

Reference is now made to Fig. 2 which is a more optimized variation of the apparatus of Fig. 1. Fig. 2 is similar to Fig. 1 except that the tables 20 and 30, subtractor 50 and positive limiter 60 are replaced by tables 100 which correspond in number to the number of existing microprocessors supported by the 7196 device. Each table 100 stores, for each instruction in the instruction register 10, the number of cycles to be wasted, typically the difference between the number of cycles required by the 7916 device and by the existing microprocessor.

The 7916 device does not have a table and therefore the table 20 input to the selector 40 in Fig. 1 is replaced by a zero input in Fig. 2.

According to a preferred embodiment of the present invention, the functions of tables 100 and selector 40 in Fig. 2 are replaced by a program based on Boolean equations representing these functions. Appendix A is a listing of a suitable program which performs the functions of tables 100 and selector 40. The program of Appendix A is written in Verilog language which is suitable for silicon design. It is appreciated that the particular embodiment described in Appendix A is intended only to provide an extremely detailed disclosure of the present invention and is not intended to be limiting.

Reference is now made to Fig. 3 which is a simplified block diagram of a peripheral unit which is a modification of the apparatus of Fig. 2. The peripheral unit of Fig. 3 is generally similar to the peripheral unit of Fig. 2. However, the apparatus of Fig. 3 additionally comprises a repeatedly programmable master clock divider 200 which is operative to divide the master clock signal governing the execution unit 74 of the associated microprocessor.

A particular feature of the present invention is that the master clock divider 200 is controlled by the execution unit 74, via clock division register 420, during operation thereof, as indicated by the "load clock divider" input signal. Provision of the peripheral unit of Fig. 3 facilitates emulation of existing microprocessors by allowing the clock division thereof to be emulated in addition to emulation of the number of machine cycles per instruction which is provided in both Figs. 2 and 3.

The advantage of the apparatus of Fig. 3 over the apparatus of Fig. 2 is apparent, for example, in applications in which a program developed for an existing microprocessor is to be mimicked and the timing of a portion of that program must be accurately mimicked. The timing of the remainder of the same program need not be accurately mimicked. The

apparatus of Fig. 3 allows the user to employ a first clock division during the "timing-accurate" portion of the program, whereas, during the remainder of the program, a second clock division may be employed which may provide other advantages such as higher speed or less current consumption at lower speed.

It is appreciated that the clock divider of Fig. 3 may, similarly, be provided in conjunction with the embodiment of Fig. 1.

The embodiments of Figs. 1 and 2 are suitable for applications in which it is sufficient to mimic the number of machine cycles and in which timing is not critical so that it is not necessary to exactly mimic the number of clock cycles.

The embodiments of Figs. 1 and 2 may be used for timing-critical applications if the machine cycle clock feeding into the cycle waster 70 is replaced by the clock signal arriving from clock 80 and tables 20, 30, of Fig. 1 and tables 100, of Fig. 2, store the number of clock cycles to be wasted.

Reference is now made to Fig. 4 which is a microprocessor constructed and operative in accordance with a preferred embodiment of the present invention. The apparatus of Fig. 4 provides the programmable clock division feature of Fig. 3, however, the cycle wasting feature of Figs. 1 and 2 are omitted.

The programmable clock division feature of Fig. 4, whether provided stand-alone or in combination with the cycle wasting feature of Figs. 1 and 2, allow a programmer to vary the processing speed of his microprocessor. This is particularly useful in applications when power consumption considerations are important. Also, when using slow external peripherals, the apparatus of Fig. 4 allows the programmer to work at a slow speed only while using the slow peripherals and to work at a higher speed otherwise.

The "load clock divider" signal in Figs. 3 and 4 may either be derived by decoding a special load instruction or by decoding a memory address in which the desired clock divider resides.

In Figs. 3 and 4, the control of the clock divider 200 by the execution unit of the microprocessor may either be direct, as illustrated in Figs. 3 and 4, or indirect, as via a suitable automaton (not shown). Also, the divided clock signal may be fed into the execution unit either directly or indirectly.

The clock division may be one-time programmable in a non-volatile register, either by the manufacturer or by the user. Alternatively, the clock division may be programmable during execution time.

Similarly, the mimicked device selector 40 of Figs. 1 and 3 may be one-time programmable in a non-volatile register, either by the manufacturer or by the user. Alternatively, the mimicked device selector may be programmable during execution time.

It is appreciated that the specific microprocessors mentioned herein are mentioned merely by-way of example and are not intended to limit the applicability of the present invention to those particular microprocessors.

It is appreciated that the execution unit may control division of the master clock signal either directly, as illustrated in Figs. 3 and 4, or indirectly, as via a suitable automaton.

Reference is now made to Fig. 5 which is a simplified block diagram of microprocessor control apparatus constructed and operative in accordance with a preferred embodiment of the present invention.

The apparatus of Fig. 5 includes an instruction register 300 receiving an individual one of a plurality of instruction codes.

An intrinsic instruction execution unit 310 contains a plurality of intrinsic instruction functions and an input 320. The execution unit 310 is operative to receive a control signal through the input 320 which designates an individual one of the intrinsic instruction functions. The designated intrinsic function is executed by the execution unit 310.

A programmable instruction mapper 330 is operative to employ a selected mapping scheme to convert the instruction code in the register into an intrinsic instruction designation which is provided as a control signal to the input 320 of the execution unit 310.

The programmable instruction mapper 330 comprises a mapping scheme depository 340 storing a multiplicity of mapping schemes and a programmable mapping scheme code register 350 which provides a control signal comprising the code of the mapping scheme which the depository is to employ.

The number of possible mapping schemes for 8-bit instruction codes is 256!. Typically, a subset of the 256! mapping schemes are provided. For example, only two mapping schemes may be provided, of which one is the "neutral" mapping scheme in which each bit is mapped onto itself and the other mapping scheme is a bit permutation followed with an inversion of some of the bits, such as the third and fifth bits.

A preferred VERILOG program for utilizing the "select mapping scheme" signal to assign a new instruction code in accordance with the selected mapping scheme is as follows:

```
always @(irx)
  being
    being
      if (!nOeMUL)
        ir = irx;
      else
        ir={irx[0],irx[3],irx[4],irx[7],irx[2],irx[5],irx[6],irx[1]}
            ^ 8'has5;
    end
  end
```

In the above sample block, the 'irx' signal arriving from the instruction register 300 is assigned to 'ir' signal when the mapping scheme is neutral. When the bit permutation mapping scheme is activated, 'irx' is permuted and some its bits are inverted by the XOR operation, designated in VERILOG with a '^'.

Fig. 6 is a simplified block diagram of microprocessor control apparatus which includes a tamper-resistant execution module 362. The tamper-resistant execution module 362 includes an execution unit 364 which performs an instruction corresponding to an instruction code arriving from an instruction register 370, via the instruction replacement module 390, whose function is described hereinbelow.

A security comparator 374 is operative to receive input from a security sensor array 380 which is operative to detect tampering attempts. The security sensor array 380 may include any suitable type of sensor such as but not limited to one or more of the following sensor types in any suitable location within the microprocessor: light detector; temperature detector; voltage-out-of-range detector, clock frequency out-of-range detector.

The security comparator 374 compares the input arriving from each sensor of the security sensor array 380 to an expected state signal representing the expected state for that sensor. The "expected state" signal may be stored in an expected state register 384 which, optionally, is associated with a non-volatile register 386 whose function is described in detail below.

For each sensor, the "expected state" signal may have only two values, corresponding to "no tampering" and "tampering detected". Optionally, the "expected state" signal may also have a third value, indicating that the input from that sensor should be enabled or disabled.

The security comparator 374 is operative to provide a "tamper detect" signal, indicating whether or not a tampering attempt has been detected, to an instruction replacement module 390. Typically the "tamper detect" signal indicates a tampering attempt if any one of the enabled sensors is not in its expected state.

The instruction replacement module 390 is enabled if a tampering attempt has been detected and is disabled otherwise. When enabled, the instruction replacement module 390 is operative to override the instruction arriving from instruction register 370 with an overriding instruction. Preferably the overriding instruction is not a load instruction so as to prevent secure information stored in microprocessor memory from traveling over the internal busses where it could be picked up by the tampering party. The overriding instruction may, for example, be a jump instruction.

According to the illustrated embodiment, the overriding instruction continues to be fed into the execution unit 364 by the instruction replacement module 390 until the security comparator 374 indicates that there is no longer any tampering attempt.

Preferably, the security comparator 374 and the instruction replacement module 390 are provided within a single module 362, as shown. Alternatively, the security comparator and/or the instruction replacement module need not be provided in the same module as execution unit 364.

A preferred VERILOG program for comparing the expected state of the security sensors to their actual state and for jamming the instruction input to the execution unit if the comparison yields a discrepancy is as follows:

```
always @(irx or security or ExpectedSecurityState)
  begin
    if (Security == ExpectedSecurityState)
      ir = irx;
    else
      ir = (irx ^ 8'h20) & 8'h20;
  end
```

The above block compares the output of security sensor 380 to the Expected Security State stored in register 384. If a match is found, the output of the instruction register 370, "irx", is mapped into an "ir" output signal which is later fed to the execution unit. In case of a mismatch, all bits of the instruction except bit 6 are forced to zero by means of the AND operation, designated "&" in VERILOG. Bit 6 is inverted by a XOR operation, designated "^" in VERILOG, resulting in two possible instruction codes 8'h00 or 8'h20 which are relative jump instructions for the 6805 microprocessor.

The "expected state" signal for security comparator 374 may be introduced into register 384 in any suitable manner, such as the following:

a. The input signal may be introduced into register 384 by means of a specific instruction performed by the execution unit;

b. The register 384 may be regarded as a memory location so that the input signal may be introduced therein by writing.

c. The register 384 may be non-volatile and the input signal may be programmed into the register by programming apparatus or by the microprocessor itself, if it is self-programming.

d. The register 384 may be volatile and an additional, non-volatile register 386 may be provided, as shown by the "dotted line" of the preferred embodiment of Fig. 6. The initial state of volatile register 384 may be determined by copying the value stored in the non-volatile register 386, either during power-up or during the reset sequence.

The registers 384 and 386 may be regarded as memory locations so as to allow the contents of register 384 to be copied into register 386 by means of standard read/write instructions, without resetting.

It is appreciated that the above methods a. - d. for introducing an input signal into a register are also applicable to the input signal for mimicked device selector 40 of Figs. 1 - 3. A suitable register and the non-volatile register required for option d are referenced 400 and 410, respectively in Figs. 1 - 3.

It is appreciated that the above methods a. - d. for introducing an input signal into a register are also applicable to the load clock divider signal for programmable clock divider 200 of Figs. 3 - 4. For this purpose, a suitable register and the non-volatile register required for option d are referenced 420 and 430, respectively in Figs. 3 - 4.

It is appreciated that the above methods a. - d. for introducing an input signal into a register are also applicable to the "select mapping scheme" signal for mapping scheme depository 340 of Fig. 5. For this purpose, a suitable register and the non-volatile register required for option d are referenced 350 and 360, respectively in Fig. 5.

Fig. 7 is a simplified block diagram of a jump unit constructed and operative in accordance with a preferred embodiment of the present invention which is operative to perform a robust conditional jump.

The apparatus of Fig. 7 includes a conditional jump unit 450 which is operative in response to an actuating "perform robust conditional jump" signal received from an instruction decoder 460.

A composite condition signal is received from a suitable source (not shown), either internal or external. The composite condition signal includes a condition component which determines the jump and an integrity component which allows the integrity of the composite condition signal to be verified. For example, the integrity component may include a copy of the condition component and/or may include the l's complement of the condition component and/or may include humming error detection code or any other suitable error detection code. The complexity of the integrity component depends on the required degree of confidence in the integrity of the composite condition signal.

The jump unit 450 is operative to perform the following operations:

a. testing the contents and the integrity of the composite condition signal within a single machine cycle;

b. jumping to a first location (go_error) if the condition signal indicates that the integrity of the condition is not maintained;

c. jumping to a second location (do_jump) if the condition signal indicates that the integrity of the condition is main-

tained and the condition is fulfilled; and

d. jumping to a third location, such as, for example, the immediately following location instruction (do_not_jump) if the condition signal indicates that the integrity of the condition is maintained and the condition is not fulfilled.

A particular feature of the apparatus of Fig. 7 is that the jump location is determined within the same cycle as the integrity check such that it is impossible for the composite condition component signal to lose its integrity after the integrity check and before the jump location is determined.

A preferred VERILOG program for performing a robust jump is as follows:

```
always @(ConditionBits or ConditionGuardBits or RequestedJump)
  begin
    DoJump = 1'h0;              //Assume no Jump
    DoNotJump = 1'h0;
    GoError = 1'h0;             //Assume no error.
    if (ConditionBits == ~ConditionGuardBits)
    begin
      if (ConditionBits[RequestedJump] == 1'h1)
        DoJump = 1'h1;
      else
        DoNotJump = 1'h1;
    end
    else
      GoError = 1'h1;
  end
```

In the above program, the condition bits are compared with the integrity component bits which is the one's complement, designated in VERILOG by the '~' operator. If the condition is intact, a requested bit is checked and a decision to Jump or not to Jump is reached. Should the condition prove not intact, a GoError signal is activated to inform the outside world of a data integrity problem and no jump is performed.

Fig. 8A is a prior art timing diagram for access to memory of any of the execution units of Figs. 1 - 6, or of the microprocessor illustrated generally in Fig. 9. The DS80C320 chip has a stretch memory cycle feature whereby the duration of the memory access signal, "access time", is programmable.

In Fig. 8A, there is no delay between the memory request signal to the memory access signal. In some state of the art multiprocessors, there is a delay (not shown) if the memory is busy due to access by another microprocessor.

Fig. 8B is a preferred timing diagram for access of any of the execution units of Figs. 1 - 6 to memory. It is appreciated that the applicability of the timing diagram of Fig. 8B is not limited to the embodiments of Figs. 1 - 6, but rather is suitable for any general microprocessor, including an execution unit 470, a memory controller 480 and a memory 490, as illustrated in prior art Fig. 9.

The timing diagram of Fig. 8B illustrates a delay between the memory request to the memory access whose duration, "delay time", is preferably pseudo-random or even random. Preferably, the range of the pseudo-random or random delay duration is programmable. Preferably, the duration of the memory access signal, "access time" is also programmable.

The memory cycle time is measured from the initiation of the memory request signal to the termination of the memory cycle completion signal supplied to the execution unit 470 by the memory controller 480 (as demonstrated by the prior art embodiment of Fig. 9). Preferably, the memory cycle time is programmable. In this case, there is typically a second delay period extending from termination of the memory access signal to the beginning of the memory cycle completion signal. The length of the second delay period is the user-selected duration of the memory cycle, minus the sum of the duration of the memory request signal, the pseudo-random/random delay time, the access time, and the duration of the memory cycle completed signal.

It is appreciated that various features of the invention which are, for clarity, described in the contexts of separate embodiments may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention is defined only by the claims that follow:

APPENDIX A

Page - 1/5	APNDX-A.DOC	14/06/94 05:18 PM

```
1   module cycles (
2     // outputs
3   //    DeltaCycles,
4       CyclesDone,
5
6     // inputs
7       SuperCycles,
8       MimicHMOS,
9       CountDown,
10      mcp,
11      ir,
12      mrsn
13  );
14  // output[2:0] DeltaCycles;
15  reg[2:0] DeltaCycles, DeltaCyclesX, DCyclesA,
16  DCyclesB;
17  output CyclesDone;
18  reg CyclesDone, PreCyDone, CyclesStart;
19
20  input SuperCycles, MimicHMOS, CountDown, mcp,
21  mrsn;
22  input [7:0] ir;
23
24  wire [3:0] IRH = ir[7:4];
25  wire [3:0] IRL = ir[3:0];
26
27  wire BTB6  = (IRH == 4'h0);
28  wire BSC3  = (IRH == 4'h1);
29  wire REL2  = (IRH == 4'h2);
30  wire DIR2  = (IRH == 4'h3);
31  wire IHA2  = (IRH == 4'h4);
32  wire IHX2  = (IRH == 4'h5);
33  wire IX13  = (IRH == 4'h6);
34  wire IX23  = (IRH == 4'h7);
35  wire IX123 = IX13 | IX23;
36
37  wire INH82 = (IRH == 4'h8);
38  wire INH90 = (IRH == 4'h9);
39  wire IMM0  = (IRH == 4'ha);
40  wire DIR1  = (IRH == 4'hb);
41  wire EXT1  = (IRH == 4'hc);
42  wire HX22  = (IRH == 4'hd);
```

```
43  wire HX12  = (IRH == 4'he);
44  wire HXX2  = (IRH == 4'hf);
45  wire DIREXT1 = DIR1 | EXT1;
46  wire HX122 = HX22 | HX12;
47  wire HX12X2 = HX122 | HXX2;
48
49  wire RTIL  = (IRL == 4'h0);
50  wire RTSL  = (IRL == 4'h1);
51  wire SWIL  = (IRL == 4'h3);
52  wire STAL  = (IRL == 4'h7);
53  wire JMPL  = (IRL == 4'hc);
54  wire RSPL  = JMPL;
55  wire TSTL  = (IRL == 4'hd);
56  wire BSRL  = TSTL;
57  wire NOPL  = TSTL;
58  wire STOPL = (IRL == 4'he);
59  wire CLRL  = (IRL == 4'hf);
60  wire STXL  = CLRL;
61  wire WAITL = CLRL;
62
63  wire TSTCLRL = TSTL | CLRL;
64  wire STASTXL = STAL | STXL;
65  wire RTIRTSL = RTIL | RTSL;
66  wire RSPNOPL = RSPL | NOPL;
67  wire SJJS = STAL | JMPL | BSRL | STXL;
68  wire SXJS = STAL | BSRL | STXL;
69
70  wire MUL0 = (ir == 8'h42);
71
72  always @(SuperCycles or ir or MimicHMOS)
73    begin
74      if (SuperCycles) DeltaCyclesX = 3'h0;
75
76      else begin
77        if (MimicHMOS)
78        begin
79          // Most instructions are one cycle more
80  efficient
81          DeltaCyclesX = 3'h1;
82
83          // 2 cycles more efficient:
```

```
84            if (REL2 || (DIR2 && !TSTCLRL) || (IHA2 &&
85        !MUL0) || IHX2 ||
86              (INH82 && (RTIRTSL || SWIL)) || (DIREXT1
87        && STASTXL) || (HX12X2 && !SJJS) ||
88              (HX122 && JMPL))
89              DeltaCyclesX = 3'h2;
90
91            // 3 cycles more efficient:
92            else if (BSC3 || (DIR2 && TSTCLRL) ||
93        (IX123 && !TSTCLRL) || (DIREXT1 && BSRL) ||
94              (HX12X2 && STASTXL) || (HXX2 && BSRL))
95              DeltaCyclesX = 3'h3;
96
97            // 4 cycles more efficient:
98            else if ((IX123 && TSTCLRL) || (IMM0 &&
99        BSRL) || (HX122 && BSRL))
100             DeltaCyclesX = 3'h4;
101
102           // 6 cycles more efficient:
103           else if (BTB6)
104             DeltaCyclesX = 3'h6;
105
106           // 0 cycles more efficient:
107           else if (MUL0 || (INH82 && (STOPL ||
108       WAITL)) || (INH90 && !(RSPL || NOPL)) ||
109             (IMM0 && !SJJS))
110             DeltaCyclesX = 3'h0;
111       end
112
113       else
114       begin // Mimic CMOS
115         // Most instructions are one cycle more
116   efficient
117           DeltaCyclesX = 3'h1;
118
119           // 2 cycles more efficient:
120           if ((DIR2 && CLRL) || (IX123 && !CLRL) ||
121   (INH82 && RTIRTSL) || (IMM0 && BSRL) ||
122             (HX12X2 && STASTXL) || (HX122 && BSRL))
123             DeltaCyclesX = 3'h2;
124
125           // 3 cycles more efficient:
```

```
126           else if (IX123 && CLRL)
127             DeltaCyclesX = 3'h3;
128
129           // 0 cycles more efficient:
130           else if ((INH90 && !RSPNOPL) || (IMM0 &&
131   !BSRL) || (DIREXT1 && !SXJS) ||
132               (HXX2 && JMPL) || MUL0 || (INH82 &&
133   (STOPL || WAITL)))
134             DeltaCyclesX = 3'h0;
135         end
136       end
137     end
138
139   always @(PreCyDone or CyclesStart)
140     begin
141       if (PreCyDone == 1'b0)
142         CyclesDone = CyclesStart;
143       else
144         CyclesDone = 1'b1;
145     end
146
147   always @(DeltaCyclesX)
148     CyclesStart = ~(|DeltaCyclesX);
149
150   always @(DeltaCycles)
151     PreCyDone = ~(|DeltaCycles);
152
153   always @(CountDown or DeltaCyclesX or DeltaCycles)
154     begin
155       if (CountDown == 1'b0)
156         DCyclesA = DeltaCyclesX;
157       else
158         DCyclesA = DeltaCycles;
159     end
160
161   always @(DCyclesA)
162     DCyclesB = DCyclesA - 1;
163
164   wire clock = (mcp & CountDown);
165
166   always @(posedge clock or negedge mrsn)
167     begin
```

```
168       if (!mrsn)
169         DeltaCycles = 3'b111;
170       else
171         DeltaCycles = DCyclesB;
172     end
173
174   endmodule
175
```

Claims

1. A microprocessor device comprising:
   an execution unit which is governed by a master clock signal; and
   a repeatedly programmable master clock divider which is operative to divide the master clock signal.

2. A microprocessor device operative to execute the instructions $I_1, ..., I_k$ executed by an existing microprocessor, said instructions being executed by the existing microprocessor in $n_1, ..., n_k$ machine cycles respectively, the microprocessor device comprising:
   an execution unit operative to execute the plurality of instructions in $m_1, ..., m_k$ machine cycles respectively, where $m_i <= n_i$ for i = 1, ..., k; and
   a machine cycle controller operative in response to a "select microprocessor" signal, to waste machine cycles such that, for each instruction, the same number of machine cycles are expended by the execution unit as by the existing microprocessor.

3. A microprocessor device according to claim 1 wherein the programmable master clock divider is one-time programmable.

4. A microprocessor device according to claim 1 wherein the programmable master clock divider is execution-time programmable.

5. A microprocessor device according to claim 2 wherein the machine cycle controller is one-time programmable.

6. A microprocessor device according to claim 1 wherein the machine cycle controller is execution-time programmable.

7. Microprocessor control apparatus comprising:
   an instruction register receiving an individual one of a plurality of instruction codes;
   an intrinsic instruction execution unit containing a plurality of intrinsic instruction functions and an input and operative to receive a control signal through the input which designates an individual one of the intrinsic instruction functions and to execute the designated intrinsic instruction function; and
   a programmable instruction mapper containing a multiplicity of mapping schemes and being operative to employ a selected scheme to convert the instruction code in the register into an intrinsic instruction designation which is provided as a control signal to the input of the execution unit.

8. A tamper-resistant execution device operative to execute instructions arriving from an instruction register, the device comprising:
   an execution unit receiving and carrying out instructions received from the instruction register;
   a tamper attempt detector operative to detect attempts to tamper with the device; and
   an instruction replacement module intermediate the instruction register and the execution unit and operative, upon receipt of a "tamper attempt detected" input from the detector, to override instructions arriving from the instruction register with other instructions.

9. A method for performing a robust conditional jump comprising:
   testing the contents and the integrity of a condition signal within a single machine cycle;

jumping to a first location if the condition signal indicates that the integrity of the condition is not maintained;

jumping to a second location if the condition signal indicates that the integrity of the condition is maintained and the condition is fulfilled; and

jumping to a third location if the condition signal indicates that the integrity of the condition is maintained and the condition is not fulfilled.

10. Memory access apparatus comprising:

a pseudo-random access delay signal generator generating an access delay signal which is at least pseudo-random and which determines the number of clock cycles to elapse from a memory access instruction clock cycle to actual memory access; and

a memory access unit receiving said access delay signal and operative to access the memory after said number of clock cycles has elapsed.

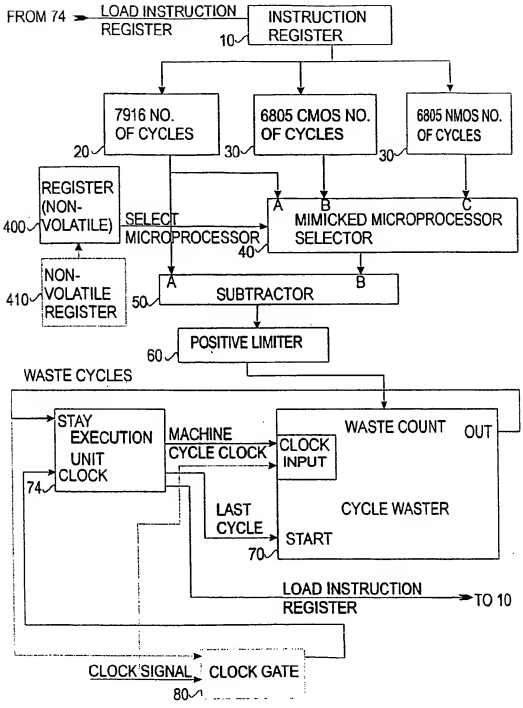

FROM 74
LOAD INSTRUCTION REGISTER
INSTRUCTION REGISTER
10
7916 NO. OF CYCLES
20
6805 CMOS NO. OF CYCLES
30
6805 NMOS NO. OF CYCLES
30
REGISTER (NON-VOLATILE)
400
SELECT MICROPROCESSOR
A
B
C
MIMICKED MICROPROCESSOR SELECTOR
40
NON-VOLATILE REGISTER
410
A
B
SUBTRACTOR
50
POSITIVE LIMITER
60
WASTE CYCLES
STAY EXECUTION UNIT CLOCK
74
MACHINE CYCLE CLOCK
CLOCK INPUT
WASTE COUNT
OUT
LAST CYCLE
START
CYCLE WASTER
70
LOAD INSTRUCTION REGISTER
TO 10
CLOCK SIGNAL
CLOCK GATE
80

Fig 1

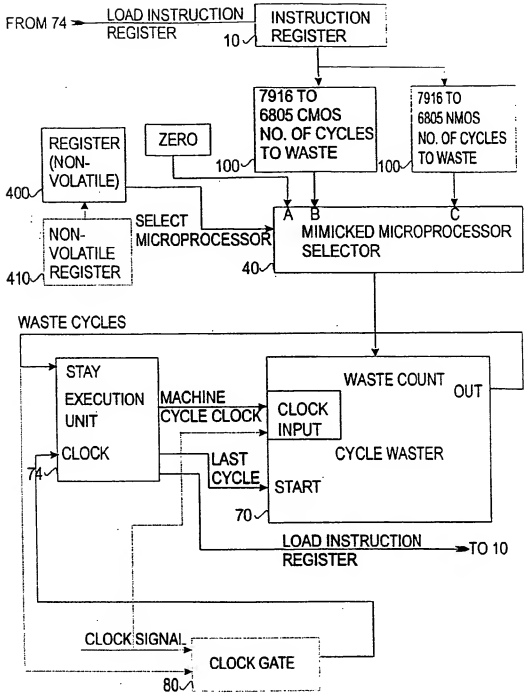

FROM 74
LOAD INSTRUCTION REGISTER
INSTRUCTION REGISTER
10
7916 TO 6805 CMOS NO. OF CYCLES TO WASTE
7916 TO 6805 NMOS NO. OF CYCLES TO WASTE
100
100
REGISTER (NON-VOLATILE)
400
ZERO
SELECT MICROPROCESSOR
A
B
C
MIMICKED MICROPROCESSOR SELECTOR
40
NON-VOLATILE REGISTER
410
WASTE CYCLES
STAY EXECUTION UNIT
CLOCK
74
MACHINE CYCLE CLOCK
CLOCK INPUT
LAST CYCLE
START
WASTE COUNT OUT
CYCLE WASTER
70
LOAD INSTRUCTION REGISTER
TO 10
CLOCK SIGNAL
CLOCK GATE
80

Fig 2

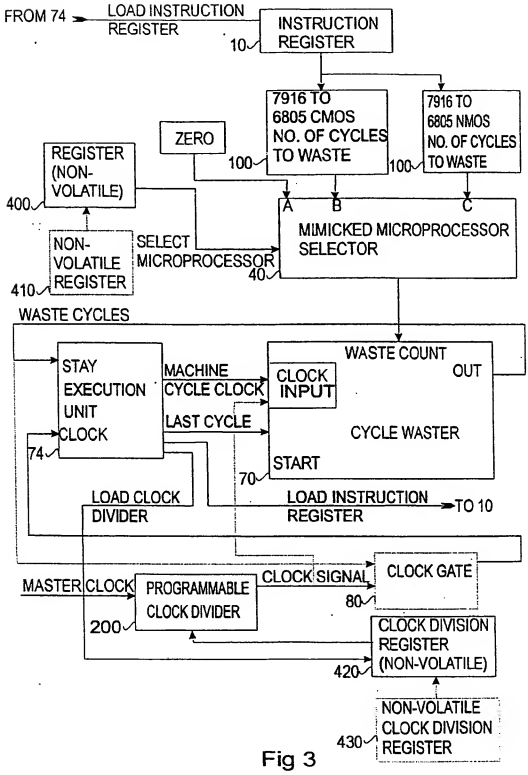

FROM 74
LOAD INSTRUCTION REGISTER
INSTRUCTION REGISTER
10
7916 TO 6805 CMOS NO. OF CYCLES TO WASTE
7916 TO 6805 NMOS NO. OF CYCLES TO WASTE
ZERO
100
100
REGISTER (NON-VOLATILE)
400
A
B
C
MIMICKED MICROPROCESSOR SELECTOR
SELECT MICROPROCESSOR
40
NON-VOLATILE REGISTER
410
WASTE CYCLES
STAY EXECUTION UNIT CLOCK
74
MACHINE CYCLE CLOCK
LAST CYCLE
WASTE COUNT
CLOCK INPUT
OUT
CYCLE WASTER
START
70
LOAD CLOCK DIVIDER
LOAD INSTRUCTION REGISTER
TO 10
MASTER CLOCK
PROGRAMMABLE CLOCK DIVIDER
200
CLOCK SIGNAL
CLOCK GATE
80
CLOCK DIVISION REGISTER (NON-VOLATILE)
420
NON-VOLATILE CLOCK DIVISION REGISTER
430

Fig 3

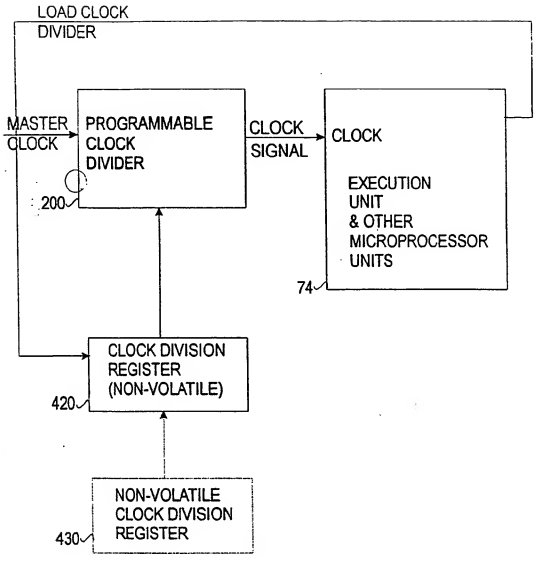

LOAD CLOCK DIVIDER
MASTER CLOCK
PROGRAMMABLE CLOCK DIVIDER
200
CLOCK SIGNAL
CLOCK
EXECUTION UNIT & OTHER MICROPROCESSOR UNITS
74
CLOCK DIVISION REGISTER (NON-VOLATILE)
420
NON-VOLATILE CLOCK DIVISION REGISTER
430

Fig 4

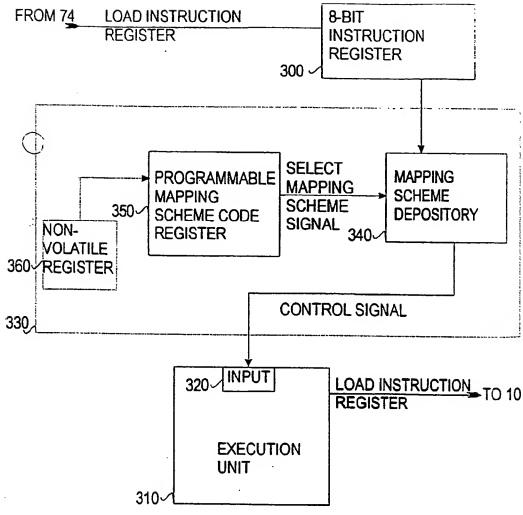

FROM 74
LOAD INSTRUCTION REGISTER
8-BIT INSTRUCTION REGISTER
300
PROGRAMMABLE MAPPING SCHEME CODE REGISTER
350
SELECT MAPPING SCHEME SIGNAL
MAPPING SCHEME DEPOSITORY
340
NON-VOLATILE REGISTER
360
CONTROL SIGNAL
330
INPUT
320
LOAD INSTRUCTION REGISTER
TO 10
EXECUTION UNIT
310

Fig 5

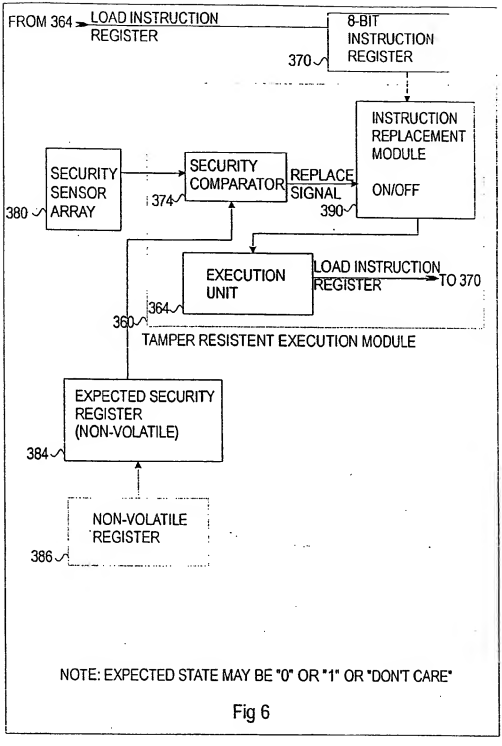

FROM 364
LOAD INSTRUCTION REGISTER
8-BIT INSTRUCTION REGISTER
370
INSTRUCTION REPLACEMENT MODULE
ON/OFF
SECURITY SENSOR ARRAY
380
SECURITY COMPARATOR
374
REPLACE SIGNAL
390
EXECUTION UNIT
LOAD INSTRUCTION REGISTER
TO 370
364
360
TAMPER RESISTENT EXECUTION MODULE
EXPECTED SECURITY REGISTER (NON-VOLATILE)
384
NON-VOLATILE REGISTER
386
NOTE: EXPECTED STATE MAY BE "0" OR "1" OR "DON'T CARE"

Fig 6

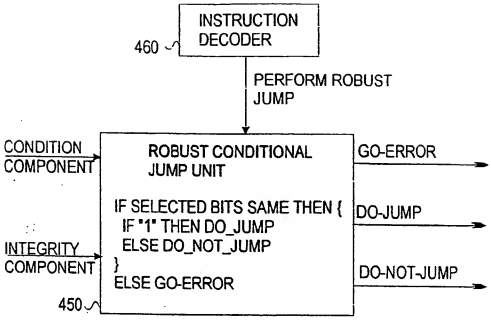
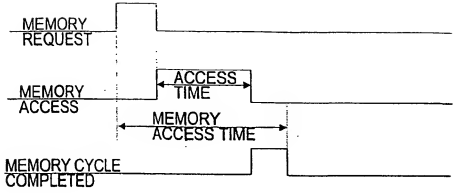

INSTRUCTION DECODER
460
PERFORM ROBUST JUMP
CONDITION COMPONENT
INTEGRITY COMPONENT
ROBUST CONDITIONAL JUMP UNIT
IF SELECTED BITS SAME THEN {
IF "1" THEN DO_JUMP
ELSE DO_NOT_JUMP
}
ELSE GO-ERROR
450
GO-ERROR
DO-JUMP
DO-NOT-JUMP

Fig 7



PRIOR ART
Fig 8A

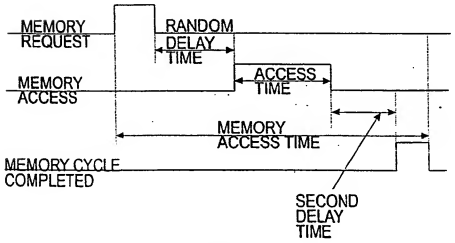
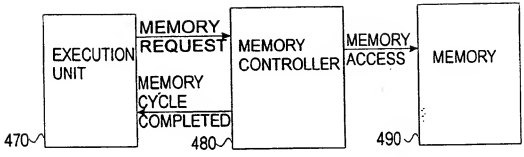


Fig 8B



PRIOR ART
Fig 9